```
001 struct Item
002 {
003     double p; // price
004     int n; // count
005
006     void IncCount( int x )
007     {
008         n += x;
009     }
010
011     void Print()
012     {
013         cout << n << "x" << p << endl;
014     }
015 };
016
017 void TestStruct()
018 {
019     Item item;
020     item.Print(); // prints uninitialized data!!!
021     item.p = 10.0;
022     item.n = 2;
023     item.Print(); // 2x10
024     item.IncCount( 5 );
025     item.Print(); // 7x10
026 }
027
028 struct Point
029 {
030     double x;
031     double y;
032
033     Point( double x0, double y0 ) :
034         x( x0 ),
035         y( y0 )
036     {
037     }
038
039     void Add( Point other )
040     {
041         x += other.x;
042         y += other.y;
043     }
044
045     void Print()
046     {
047         cout << x << " " << y << endl;
048     }
049 };
050
051 class Line
052 {
053     public:
054         Line( double x1, double y1, double x2, double y2 ) :
055             p1_( x1, y1 ),
056             p2_( x2, y2 )
057         {
058         }
059         Line( Point p1, Point p2 ) :
060             p1_( p1.x, p1.y ),
061             p2_( p2.x, p2.y )
062         {
063         }
064
065         double GetLength()
066         {
067             return sqrt((p1_.x - p2_.x) * (p1_.x - p2_.x) +
068                         (p1_.y - p2_.y) * (p1_.y - p2_.y));
069         }
070
071         void Print()
072         {
073             p1_.Print();
074             p2_.Print();
075         }
076
077     private:
078         Point p1_;
079         Point p2_;
080 };
```

```
081
082 void TestClass()
083 {
084     Line l1( 1.0, 0.0, 10.0, 0.0 );
085     cout << l1.GetLength() << endl; // 9
086     Point p1( 2.0, 0.0 );
087     Point p2( 5.0, 0.0 );
088     Line l2( p1, p2 );
089     l2.Print(); // 2 0 \n 5 0
090     Line l3( Point( 1.0, 2.0 ), Point( 3.0, 4.0 ) );
091     l3.Print(); // 1 2 \n 3 4
092 }
093
094 class Figure
095 {
096     public:
097         Figure( Point p ) :
098             center_( p )
099         {
100         }
101
102     protected:
103         Point center_;
104 };
105
106 class Square : public Figure
107 {
108     public:
109         Square( Point center, double side ) :
110             Figure( center ),
111             side_( side )
112         {
113         }
114
115         void Print()
116         {
117             cout << side_ << " at " << center_.x << ", " <<
center_.y << endl;
118         }
119
120     protected:
121         double side_;
122 };
123
124 class DynamicSquare : public Square
125 {
126     public:
127         DynamicSquare( Point center, double side ) :
128             Square( center, side )
129         {
130         }
131
132         void Move( Point d )
133         {
134             center_.Add( d );
135         }
136         void Move( double dx, double dy )
137         {
138             center_.Add( Point( dx, dy ) );
139         }
140     private:
141 };
142
143 void TestInheritance()
144 {
145     Square s1( Point( 1.0, 2.0 ), 5.0 );
146     s1.Print(); // 5 at 1, 2
147     DynamicSquare s2( Point( 1.0, 1.0 ), 2.0 );
148     s2.Move( Point( 1.0, 2.0 ) );
149     s2.Print(); // 2 at 2, 3
150     s2.Move( 1.0, 2.0 );
151     s2.Print(); // 2 at 3, 5
152 }
```

```
01 #include <iostream>
02 #include "AllegroUtil.hpp"
03 #include <windows.h>
04 #include <cstdlib>
05
06 using namespace std;
07
08 const int FPS = 60;
09 const int SCREEN_W = 640;
10 const int SCREEN_H = 480;
11
12 int i = 0;
13 void draw1()
14 {
15     ++i;
16     cout << "frame " << i << endl;
17
18     al_clear_to_color( al_map_rgb( 0, 0, 0 ) );
19     al_put_pixel( 50, 50, al_map_rgb( 0, 255, 0 ) );
20     al_draw_line( 100, 100, 300, 200,
                     al_map_rgb( 255, 0, 0 ), 5 );
21     al_draw_triangle( 120, 120, 150, 120, 130, 150,
                         al_map_rgb( 255, 0, 0 ), 3 );
22     al_draw_filled_triangle( 120, 220, 150, 220, 130, 250,
                                al_map_rgb( 0, 255, 0 ) );
23     al_draw_rectangle( 300, 300, 350, 350,
                          al_map_rgb( 0, 255, 0 ), 1 );
24     al_draw_filled_rectangle( 350, 300, 400, 350,
                                 al_map_rgb( 0, 255, 255 ) );
25     al_draw_circle( 500, 400, 50, al_map_rgb(0, 255, 0), 3 );
26     al_draw_filled_circle(400,400,50, al_map_rgb(0, 255,0) );
27 }
28
29 void draw2()
30 {
31     for( int i = 0; i < 10; ++i )
32     {
33         al_draw_line( 100, 100, 200 + i * 10, 200,
                         al_map_rgb( 255, 0, 0 ), 2 );
34         al_flip_display();
35         Sleep( 1000 ); // one second sleep, from <windows.h>
36     }
37     ExitAllegro();
38 }
39
40 struct Circle
41 {
42     double x;
43     double y;
44     double dx;
45     double dy;
46     double r;
47     void Reset()
48     {
49         x = SCREEN_W / 2;
50         y = SCREEN_H / 2;
51         r = 10.0 + rand() % 100;
52         dx = 10.0 - rand() % 21;
53         dy = 10.0 - rand() % 21;
54     }
55 };
56
57 Circle circle;
58 void fps3()
59 {
60     circle.x += circle.dx;
61     circle.y += circle.dy;
62     if ( ( circle.x < 1.0 ) ||
63          ( circle.x > SCREEN_W ) ||
64          ( circle.y < 1.0 ) ||
65          ( circle.y > SCREEN_H ) )
66     {
67         circle.Reset();
68     }
69 }
70
71 void draw3()
72 {
73     al_clear_to_color( al_map_rgb( 0, 0, 0 ) );
74     al_draw_filled_circle( circle.x, circle.y, circle.r,
                              al_map_rgb( 0, 255, 0 ) );
75 }
```

```
76
77 int main(int argc, char **argv)
78 {
79     srand( time(0) );
80     if( !InitAllegro( SCREEN_W, SCREEN_H, FPS ) )
81     {
82         DestroyAllegro();
83         return 1;
84     }
85
86     //RunAllegro( 0, &draw1 );
87     //RunAllegro( 0, &draw2 );
88     circle.reset();
89     RunAllegro( &fps3, &draw3 );
90
91     DestroyAllegro();
92     // cin.get();
93     return 0;
94 }
```

```
Check compiler:
settings -> compiler -> GNU GCC Compiler
Tollchain Executable == "c:\CodeBlocks16\MinGW
Check debugger:
Settings -> debugger
Default: Executable path == c:\CodeBlocks16\MinGW\bin\gdb.exe
```

```
(right click on project)Build options:
1)Search directories -> Compiler
C:\codeblocks\MinGW\include
2)Linker Settings
C:\codeblocks\MinGW\lib\liballegro.dll.a
C:\codeblocks\MinGW\lib\liballegro_primitives.dll.a
3) если линковщик ругеается, то ещё
Search directories -> Linker
C:\codeblocks\MinGW\bin

Copy to root AllegroUtil.cpp/hpp and add files(right click on
project)

To run standalone windows need to find DLLs:
C:\codeblocks\MinGW\bin\allegro-5.0.dll
C:\codeblocks\MinGW\bin\allegro_primitives-5.0.dll
Add to %PATH% or copy to executable folder(..your_project\bin\
Debug)
Help: C:\codeblocks\MinGW\allegro\docs\html\refman\index_all.html

ВНИМАНИЕ! Имя Rectangle кофликтует с windows.h => использовать
что-то другое. Например, Rect

Google style guide:
https://google.github.io/styleguide/cppguide.html
```